Evaluation of Struggle Strategy in Genetic Algorithms for Ground Stations Scheduling Problem

Fatos Xhafa[a,*], Xavier Herrero[a], Admir Barolli[b], Leonard Barolli[c], Makoto Takizawa[b]

[a] *Technical University of Catalonia, Barcelona, Spain.*
[b] *Seikei University, Tokyo, Japan.*
[c] *Fukuoka Institute of Technology, Fukuoka, Japan.*

Abstract

Ground station scheduling problem arises in spacecraft operations and aims to allocate ground stations to spacecraft to make possible the communication between operations teams and spacecraft systems. The problem belongs to the family of satellite scheduling for the specific case of mapping communications to ground stations. The allocation of a ground station to a mission (e.g. telemetry, tracking information, etc.) has a high cost, and automation of the process provides many benefits not only in terms of management, but in economic terms as well. The problem is known for its high complexity as it is an over-constrained problem. In this paper, we present the resolution of the problem through Struggle Genetic Algorithms –a version of GAs that distinguishes for its efficiency in maintaining the diversity of the population during genetic evolution. We present some computational results obtained with Struggle GA using the STK simulation toolkit, which showed the efficiency of the method in solving the problem.

Keywords: Ground station scheduling, Satellite scheduling, Struggle Genetic Algorithms, Constraint programming, Simulation.

*Corresponding author.
Email addresses: fatos@lsi.upc.edu (Fatos Xhafa), xherrero@lsi.upc.edu (Xavier Herrero), admir.barolli@gmail.com (Admir Barolli), barolli@fit.ac.jp (Leonard Barolli), makoto.takizawa@st.seikei.ac.jp (Makoto Takizawa)

1. Introduction

Ground Station Scheduling is one of the most important problems in the field of Satellite-Scheduling. It consists in computing feasible planning of communications between satellites or spacecraft (SC) and operations teams of Ground Station (GS). The problem arises in many real life applications and projects, such as from ESA (European Space Agency) [7, 6, 2] (see Fig. 1 for ESA Tracking Network) and NASA [5]. In fact, there is a growing interest and need of the efficient resolution of the problem also from smaller projects from research institutions and universities.

Ground Station Scheduling is a very complex problem due to its over-constrained nature. Indeed, there are several restrictions that make the planning of even small problem instances too complex to deal with manually or by brute force search algorithms.

- *The over-constrained nature:* There is a large set of constraints. In fact, this is the first major difference between the problems of conventional scheduling and that of Ground Station scheduling. First, there are restrictions on the communication time required for each SC in a period of time. Secondly, there are restrictions on the visibility of each window on each Spacecraft Ground Station, i.e. the time at which each SC can communicate with each GS in a given time period. Resources are thus not available at all times for mission allocation.

- *Communication time variables:* The second major difference with other planning algorithms is that in the case of Ground-Station Scheduling, the length of the communication is variable, where it should be at least the required communication time and at most the maximum time within which the window visibility ends or the visibility window of another communication starts.

All scheduling variants, in their general formulations, are highly constrained problems and have been shown computationally hard [11, 9, 3, 13].

In this paper, we propose the resolution of Ground-Station Scheduling using Struggle Genetic Algorithms. Struggle GA is a version of GAs, which aims to keep a diverse population of individuals. For that, Struggle GA creates a new generation by replacing only a portion of the population with newly generated individuals. A new individual replaces the individual that is *genetically* most similar to it only in case the new individual obtains a better

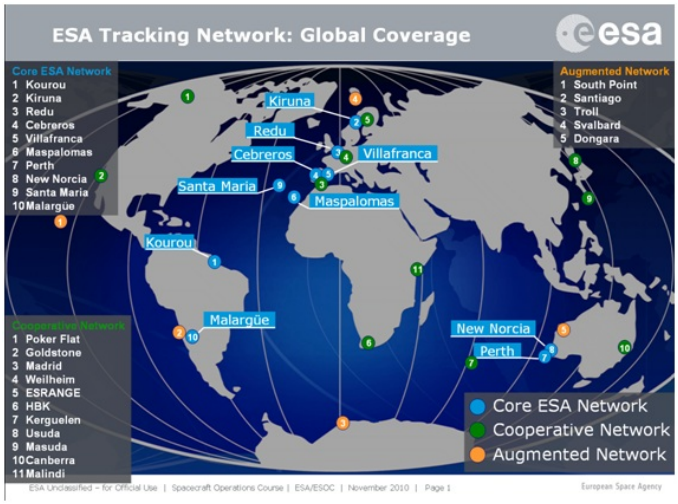

Figure 1: ESA Tracking Network.

fitness value than the one to be replaced. The aim is to avoid premature convergence and reach a better convergence point. Several similarity functions for individuals are considered (Hamming, Euclidean, Cosine and Hash-based functions) to identify the one that works best for the problem We have experimentally evaluated the Struggle GA using the STK simulation toolkit, by generating a family of instances of different sizes (small, medium, large) that tries to capture real features of the problem.

The rest of the paper is organized as follows. In Section 2 we describe the Ground-Station Scheduling. The different fitness types for the problem are formulated in Section 3. The Struggle GA is given in Section 4 and its experimental evaluation in Section 5. We end the paper in Section 6 with some conclusions and remarks for future work.

2. The Ground-Station Scheduling Problem

Problem input instance. The input instance is defined in Table 1.

Objectives. Different types of objectives can be formulated, namely, maximizing matching of visibility windows of spacecrafts to communicate with ground stations, minimizing the clashes of different spacecrafts to one ground station, maximizing the communication time of spacecraft with ground station, and maximizing the usage of ground stations. The challenge here is to optimize several objectives.

Table 1: Parameters defining the input instance

Parameter	Description
$SC\{i\}$	List of Spacecrafts that participate in the planning
$GS\{g\}$	List of Ground Stations that participate in the planning
N_{days}	Number of days for the schedule
$TAOS\_VIS(i)(g)$	Visibility time of GS to SC
$TLOS\_VIS(i)(g)$	Time from which a GS looses signal from SC
$TReq(i)$	Communication time required for spacecrafts

Constraints. The most common constraint in Ground Station Scheduling is the clash of visibility windows caused by multiple spacecrafts willing to communicate to a ground station (see Table 2 for an example of time requirements for three spacecrafts).

Table 2: Time Requirements for spacecrafts.

SC	From (min)	To (min)	Require (min)	Meaning
1	1	2880	60	1 hour / 2 days
1	2881	5760	60	1 hour / 2 days
1	5761	8640	60	1 hour / 2 days
1	8641	12960	60	1 hour / 2 days
2	1	2880	80	80 mins / 2 days
2	2881	5760	80	80 mins / 2 days
2	5761	8640	80	80 mins / 2 days
2	8641	12960	80	80 mins / 2 days
3	1	1440	120	2 hours / day
3	1441	2880	120	2 hours / day
3	2881	4320	120	2 hours / day
3	4321	5760	120	2 hours / day
3	5761	7200	120	2 hours / day
3	7201	8640	120	2 hours / day

As an example, the data for the time windows of SC/GS is defined as in Table 3.

Table 3: Resuming data specification for time windows

GS	SC	AOS-VIS	LOS-VIS	TDur
1	1	08-FEB-2012. 12:00:00	08-FEB-2012. 14:00:00	120 min

Problem output. A solution procedure to the problem should output the values of the parameters defined in Table 4.

Table 4: Parameters defining the problem output

Parameter	Description
$T_{Start}(i)(g)$	Starting time of the communication of a $SC(i)$ with a $GS(g)$
$T_{Dur}(i)(g)$	Duration time of the communication of a $SC(i)$ with a $GS(g)$
$SC\_GS(i)$	The GS assigned to every $SC(i)$.
$Fit_{LessClash}$	The fitness of minimizing the collision of two or more SC to the same GS for a given time period (measured from 0 to 100).
$Fit_{TimeWin}$	The fitness value corresponding to time access window for every pair $GS - SC$ (measured from 0 to 100).
Fit_{Req}	Fitness value corresponding to satisfying the requirement on the mission communication time (measured from 0 to 100).
Fit_{GSU}	Fitness value corresponding to maximizing the usage of all GS during the planning (measured from 0 to 100).

3. Scheduling fitness types

One of the major complexities of the mission operations scheduling comes from the many objectives that can be sought for the problem. These objectives are related to visibility window, communication clashes, communication time and ground station resource usage, among others. The total fitness function, besides being composed of multiple objectives, poses the challenge of how to combine them and in which order to evaluate them. For the combination, one can adopt a hierarchical optimization approach based on the priority of the objectives or a simultaneous optimization approach. In the former, objectives are sorted according to some priority criteria and are optimized according that ordering. In the later, objectives are simultaneously optimized, e.g. by summing up all fitness functions into one single fitness function.

We define next the four main objectives that would compose the fitness function.

3.1. Access window fitness

Visibility windows are the time periods when a GS has the possibility to setup a communication link with a SC. The objective is that all or the largest possible number of generated communication links to fall into access windows and thus achieve as many communications as possible. In the following equation, $W_{(g,i)}$ is the Access Window set for Ground Station g and Spacecraft i, $T_{Start}(s)$ and $T_{End}(s)$ are the start and end of each access window.

$$AW(g,i) = \cup_{s=1}^{S}[T_{AOS(g,i)}(s), T_{LOS(g,i)}(s)]$$

Then, we define the final Access Window fitness of the scheduling solution (Fit_{AW}) calculated as follows:

$$f(n) = \begin{cases} 1, & \text{if } [T_{Start}(n), T_{Start}(n) + T_{Dur}(n)] \subseteq AW(n_g, n_i), \\ 0, & \text{otherwise.} \end{cases}$$

$$Fit_{AW} = \frac{\sum_{n=1}^{N} f(n)}{N} * 100,$$

where n value corresponds to an event, N is the total number of events of an entire schedule, g is a ground station and i a spacecraft (see Fig. 2). The fitness of access window is normalized so that it's value is within 0 to 100.

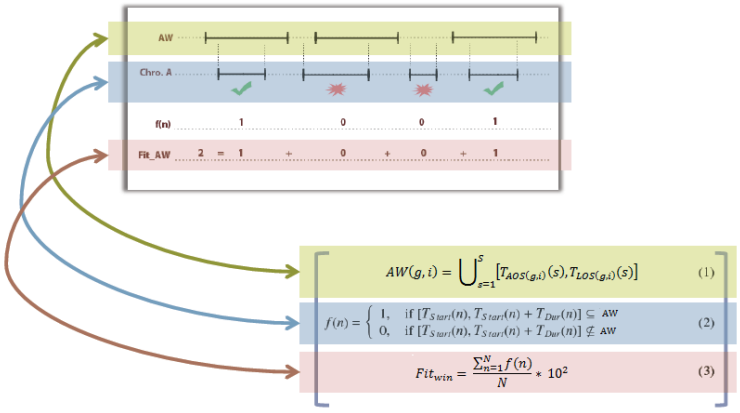

Figure 2: Access Window Fitness.

3.2. Communication clashes fitness

Communications clash represents the event when the start of one communication task happens before the end of another one on the same ground station. The objective is to minimize the clashes of different spacecrafts to one ground station. To compute the number of clashes, SCs are sorted by their start time. If, as a result of the sorting:

$$T_{Start}(n+1) < T_{Start}(n) + T_{Dur}(n), \quad 1 \leq n \leq N-1$$

where n value corresponds to an event and N is the total number of events of an entire schedule, then there is a clash. The fitness will be reduced, and

one of the clashed entries has to be removed from the solution (see Fig. 3 for an example). The total fitness of communication clashes is then:

$$f(n) = \begin{cases} -1, & \text{if } T_{Start}(n+1) < T_{Start}(n) + T_{Dur}(n), \\ 0 & \text{otherwise.} \end{cases}$$

$$Fit_{CS} = \frac{N + \sum_{n=1}^{N} f(n)}{N}$$

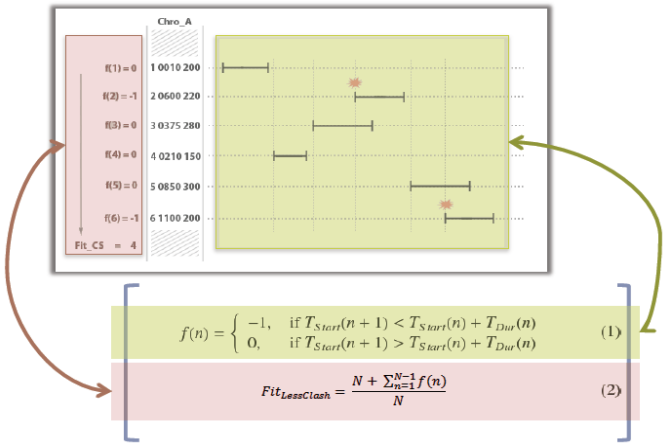

Figure 3: Fitness communication clashes.

3.3. Communication time requirement fitness

The objective is to maximize the communication time of spacecrafts with ground stations so that every spacecraft $SC(i)$ will communicate at least $T_{req}(i)$ time. Thus, a sufficient amount of time should be granted for TTC (Telemetry, Tracking and Command). For example, satellites that need to download huge amount of image data require more time for linking with ground stations. These communications, especially for data download tasks are usually periodical tasks (e.g. 2 hours communication for SC1 each day, 5 hours data downlink for SC2 every 2 days, etc.) A matrix is used to define those requirements, which is used as the input for the scheduling system (see Fig. 4 for an example.)

The fitness is calculated by summing up all the communication link durations of each spacecraft, and dividing them in the required period to compare if the scheduled time matches requirements (see Eqs. (1) and also Fig 5).

$$
\begin{array}{cccc}
SC & From & To & T_{REQ}[Min] \\
\left[\begin{array}{cccc}
SC[1] & T_{From}[1]_1 & T_{TO}[1]_1 & T_{REQ}[1]_1 \\
SC[1] & T_{From}[1]_2 & T_{TO}[1]_2 & T_{REQ}[1]_2 \\
\vdots & & & \\
SC[1] & T_{From}[1]_N & T_{TO}[1]_N & T_{REQ}[1]_N \\
SC[2] & T_{From}[2]_1 & T_{TO}[2]_1 & T_{REQ}[2]_1 \\
SC[2] & T_{From}[2]_2 & T_{TO}[2]_2 & T_{REQ}[2]_2 \\
\vdots & & & \\
SC[i] & T_{From}[i]_n & T_{TO}[i]_n & T_{REQ}[i]_n \\
\vdots & & & \\
SC[I] & T_{From}[I]_N & T_{TO}[I]_N & T_{REQ}[I]_N
\end{array}\right]
\end{array}
$$

Figure 4: Matrix representation of periodic tasks.

$$
\begin{aligned}
T_{Start}(m) &> T_{From}(k) \\
T_{Start}(n) + T_{Dur}(n) &< T_{TO}(k) \\
T_{Comm}(k) &= T_{Dur}(j) \qquad (1) \\
f(k) = \begin{cases} 1, & \text{if } T_{Comm}(k) \geq T_{REQ}(k), \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

$$
FIT_{TR} = \frac{\sum_{k=1}^{K} f(k)}{N} \cdot 100.
$$

3.4. Ground station usage fitness

Given that the number of ground stations is usually smaller than the number of spacecrafts missions, the objective is to maximize the usage of ground stations, that is, try to reduce the idle time of a ground station. A maximized usage would contribute to provide additional time for SC communications (see Fig. 6 for an example).

This fitness value is calculated as the percentage of ground stations occupied time by the total amount of the possible communication time. The more a GS is used, the better is the corresponding schedule.

$$
Fit_{GU} = \frac{\sum_{n=1}^{N} T_{Dur}(n)}{\sum_{g=1}^{G} T_{Total}(g)} \cdot 100.
$$

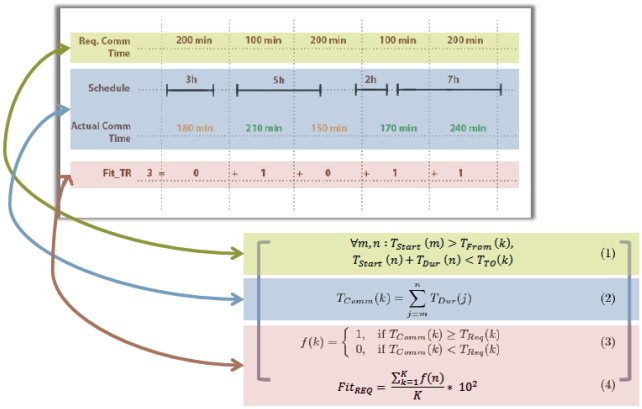

Figure 5: Fitness Requirements .

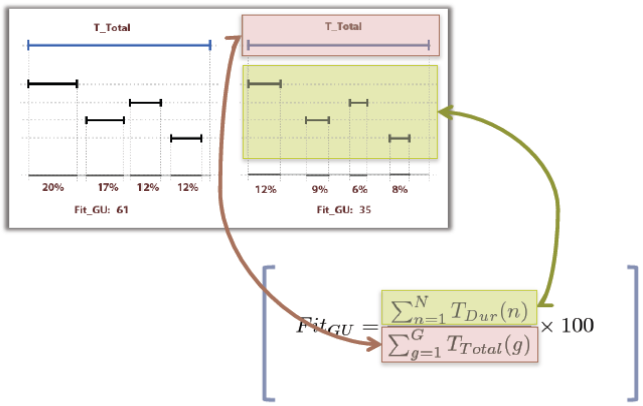

Figure 6: Ground station usage.

where N is the number of events of an entire schedule, G is the number of ground stations and $T_{Total(g)}$ is the total available time of a ground station

3.5. Combination of fitness objectives

The fitness objectives defined above (FIT_{AW}, FIT_{CS}, FIT_{TR}, FIT_{GU}) are conceived as fitness modules so as to facilitate the design phase of the scheduler to easily plug-in other fitness objectives. From the definition of the fitness objectives, we can observe that some of them can be applied in serial fashion (due dependencies, denoted serial-FM), while some others can be applied in parallel (denoted parallel-FM). Thus, in a hierarchical mode, one possible way to arrange fitness checking is that of Fig. 7.

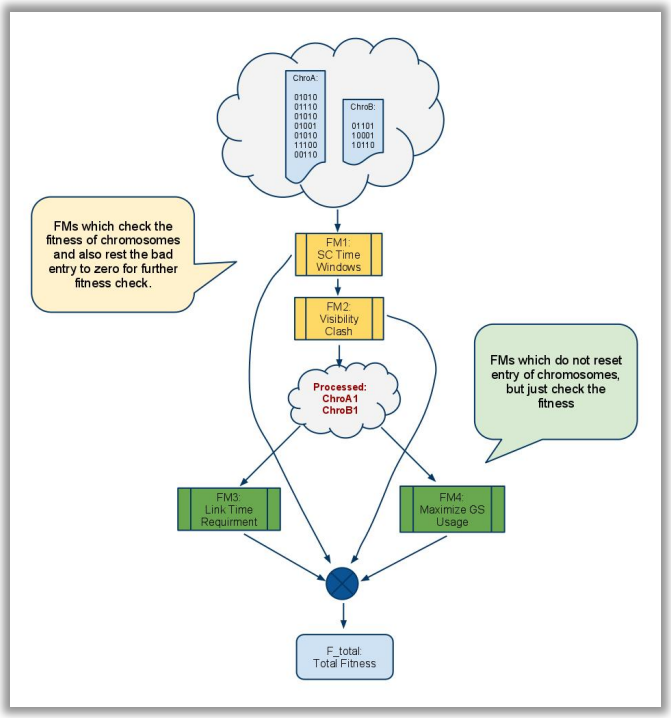

Figure 7: Combination of fitness modules in hierarchic mode.

With regard to the simultaneous combination of objectives, one can either consider a proper Pareto-front approach, or combine all the fitness modules into one total fitness function using weights for different fitness module:

$$Fit = \sum_{i=1}^{n} w_i \cdot Fit_S(i) + \sum_{j=1}^{m} w_j \cdot Fit_P(j)$$

where w_i, w_j are the weights of fitness modules, $Fit_S(i)$ and $Fit_P(j)$ are the fitness values from Serial-FMs and Parallel-FMs, and n, m are the number of fitness modules, resp. More precisely, we define the total fitness function as follows:

$$Fit_{TOT} = \lambda \cdot Fit_{Win} + Fit_{Req} + \frac{Fit_{LessClash}}{10} + \frac{Fit_{GSU}}{100}.$$

for some λ (defined to $\lambda = 1.5$ for the experimental study).

4. Genetic Algorithm for ground station scheduling

Several heuristics algorithms can be used to search for near-optimal solutions to the problem. Some approaches proposed in the literature include Genitor [4], Branch-and-Bound Algorithm [4], Graph Colouring [13], Tabu Search [10], Hill Climbing [4], Fuzzy techniques [1]. We have used the *GA template* of Alg. 1 in this study.

Algorithm 1. *Struggle Genetic Algorithm*

Generate the initial population P^0 *of size* μ*;*
Evaluate P^0*;*
while *not termination-condition* ***do***
  Select the parental pool T^t *of size* λ*;* $T^t := Select(P^t)$*;*
  Perform crossover procedure on pairs of individuals in T^t *with probability* p_c*;*
  $P_c^t := Cross(T^t)$*;*
  Perform mutation procedure on individuals in P_c^t *with probability* p_m*;*
  $P_m^t := Mutate(P_c^t)$*;*
  Evaluate P_m^t*;*
  Create a new population P^{t+1} *of size* μ *from individuals in* P^t *and/or* P_m^t*;*
  $P^{t+1} :=$ `StruggleReplace`$(P^t; P_m^t)$
  $t := t + 1$*;*
end while
return *Best found individual as solution;*

end

4.1. GA initial population

The generation of individuals of the first population is done at random, however, a few individuals were generated using some *ad hoc* solutions (see Fig. 8) aiming to introduce more diversity to the population.

- *Random First*: This method generates a solution with time intervals situated in the first half day of everyday in the specified period, that is:

$$N_d \in (0..N_{days} - 1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$

$$T_{Start}[i] = random(1, \frac{MINPERDAY}{2}) + day * MINPERDAY$$

  where N_{SC} is the number of Spacecrafts, $MINPERDAY$ is a constant that indicates the amount of minutes per day.

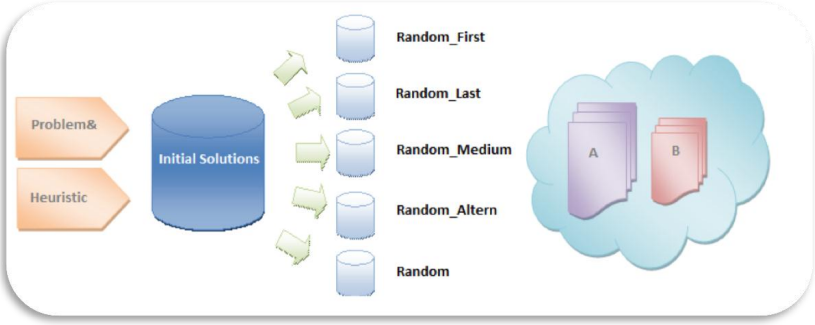

Figure 8: Generation of individuals of the first population.

- *Random Last*: This method generates a solution with time intervals situated in the second half day of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(\frac{MINPERDAY}{2}, MINPERDAY - 1) + \\ + day * MINPERDAY$$

- *Random Medium*: This method generates a solution with time intervals situated from one third to two third interval of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(\frac{MINPERDAY}{3}, \frac{2 * MINPERDAY}{3} + \\ + day * MINPERDAY$$

- *Random Altern*: This method generates the intervals in even position using the Random First and those in odd position using Random Last.

- *Random*: This method generates the intervals at random in the full available time of everyday in the specified period, that is:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Start}[i] = random(1, MINPERDAY - 1) + day * MINPERDAY$$

Finally, the values of $T_{Dur}[i]$ are generated based on the previously computed values assigned to T_{Start}, as follows:

$$N_d \in (0..N_{days}-1),\ N_d = \lfloor \frac{i}{N_{SC}} \rfloor,\ MINPERDAY = 1440$$
$$T_{Dur}[i] = random(1, MINPERDAY * (day + 1) - T_{Start}[i]) + \\ + day * MINPERDAY$$

4.2. GA convergence

A key issue in GA design is the convergence of the algorithm, namely, a fast convergence of the population would stagnate the search to local optima whereas slower convergence would require a considerably longer time towards sub-optimal solutions. The convergence of GAs is determined by selection and replacement strategies. The selective pressure directly affects the tradeoff between the exploration and exploitation of the search space. Indeed, if the population converges rapidly GA would give more priority to the exploitation and, vice-versa, when the population is kept diverse, other regions of the search space would be explored aspiring thus to find better solutions.

4.3. Struggle GA replacement

We focus here in using the Struggle strategy [8] (`StruggleReplace` in Alg. 1). In Struggle GA, a new individual replaces the individual that is most similar to it only in case the new individual obtains a better fitness value than the one to be replaced. This strategy is known for its effectiveness but suffers from a high computational cost. More precisely, given a new individual, finding a similar individual to it requires comparing against all individuals of the current generation. The definition of effective similarity functions is therefore crucial to Struggle GA.

4.3.1. Standard similarity measures

In order to compare the similarity between solutions, a measure of similarity or distance function has to be defined. Standard similarity measures include:

- *Hamming distance*: given two individuals S_1 and S_2 encoding two scheduling of N tasks, let $g[i] = 1$, iff $S_1[i] = S_2[i]$ and $g[0] = 0$, otherwise. Similarity is then calculated as:

$$Sim_h(S_1, S_2) = \frac{\sum_{i=1}^{N} g[i]}{N}.$$

- *Euclidean distance*: This similarity is based on the Euclidean distance. Given two solutions S_1 and S_2, by considering them as two points in N-dimensional space, the similarity is then computed as the Euclidean distance between them:

$$Sim_e(S_1, S_2) = \sqrt{\sum_{i=1}^{N} (S_1[i] - S_2[i])^2}.$$

- *Cosine distance*: In this case, the similarity is measured using the angle of the two vector solutions S_1 and S_2 of the N-dimensional space. Cosine values close to 1 would mean higher similarity.

$$Sim_c(S_1, S_2) = \frac{\sum_{i=1}^{N} S_1[i] \cdot S_2[i]}{\sqrt{\sum_{i=1}^{N} S_1^2[i]} \cdot \sqrt{\sum_{i=1}^{N} S_2^2[i]}}.$$

4.3.2. Hash-based similarity measure

The standard similarity measures given above have linear time computational cost in number of tasks. Therefore for a population of *pop_size* the standard struggle strategies would take $O(intermediate\_pop\_size \times pop\_size) \times N$, where N is the number of tasks.

Reducing the quadratic factor of $O(intermediate\_pop\_size \times pop\_size)$ to a linear time factor would be very desirable in this case since in each replacement step it would take a considerable time in detriment to the proper search time of the GA. In order to achieve this, we propose the use of hash techniques so that given a new individual of the intermediate population

we can find in constant time the individual in the current population most similar to it.

In order to design the hash table, we have to first define the key to identify the individuals of the population. The *key* information is the basis for computing the degree of similarity of the *struggle genetic operator*: the more accurate its definition, the better the performance of the operator. In fact, a poor definition of the key would simply reduce the struggle operator to a random replacement. In our definition of the key the context is crucial: the key value should resume as much as possible the genetic information encoded in an individual; hence, if two key values are similar then their respective individuals are genetically similar. The following are three possible definitions:

a) *Fitness-based key*: consists in using the fitness value, which is transformed, using a hash function, into the key value. We refer to this as `'a' key`.

b) *Position-based key*: having the permutation vector of task-resource allocation, in which tasks are sorted according to the resource they are assigned to, the key is defined as the sum of number of cells a component of the vector would move to the right as indicated by its value, when the vector is read in a circular way (we refer to this as `'b' key`).

For instance, for the vector of 7 tasks in Fig. 9 below, $key = 2 + 4 + 1 + 0 + 2 + 5 + 0 = 14$.

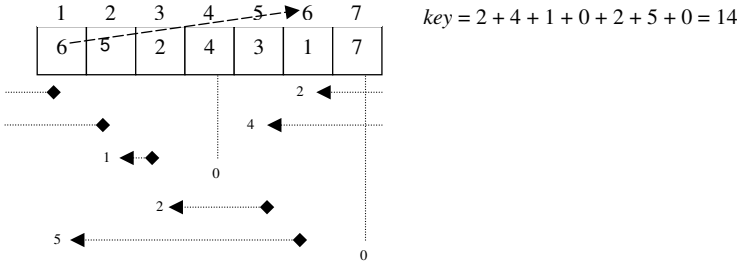

Figure 9: Example of position-based key calculation.

Note that this definition uses the genetic characteristics of the solution; however, the relation task-resource is not explicitly taken into account, i.e., to which resource a task is assigned to.

c) *Task-resource allocation key*: In this case both information on tasks and ground stations is used. The key value is now the sum of the absolute values of the subtraction of each position and its precedent in the vector of task-ground station allocation (reading the vector in a circular way); we refer to this as `'c' key`.

We give in Fig. 10 the graphical representation of the hash table design as well as the formulae definition of the hash function. Note that the corresponding position is obtained from a solution from the key value k; k_{min} and k_{max} correspond respectively to the key with smallest and largest value in the population.

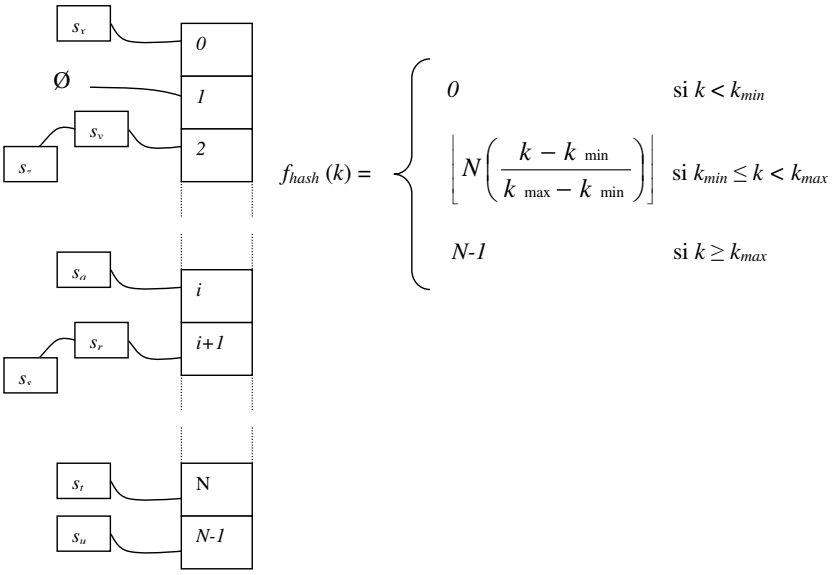

Figure 10: Representation of the hash table and the hash function definition.

The hash table has the same size as the population in order to obtain constant time access (in average). If an access fails, a few individuals in the population are randomly chosen and the most similar to the new one is considered for the replacement. Hence, the constant access is always ensured even if a failed access occurs. Therefore, the computational cost of the hash-based struggle operator is linear time $O(pop\_size + intermediate\_pop\_size)$.

4.4. Chromosome encoding and GA operators

For the chromosome encoding, initial population and GA operators (crossover, mutation, selection) see [14, 15].

5. Computational results

5.1. Problem instances

We present some computational results obtained with Struggle GA for the ground stations scheduling problem. The Satellite Tool Kit (STK) [12] is used to generate some simulation scenarios of small, medium and large sizes, described in Table 5.

Table 5: Different size instances description

Small size Instances	
Number of Ground Stations	5
Spacecrafts number	10
Number of days	10
Medium size instances	
Number of Ground Stations	10
Spacecrafts number	15
Number of days	10
Large size instances	
Number of Ground Stations	15
Spacecrafts number	20
Number of days	10

5.2. GA parameters

A total of 20 independent runs of the Struggle GA were performed (under the same parameter configuration –see Table 6) and average results are reported.

Table 6: Struggle GA parameter values

Parameter	Value
Number of evolution steps	30000
Crosspoint chromosome A	10
Crosspoint chromosome B	3
Mutation rate chromosome A	15%
Mutation rate chromosome B	5%
Crossover probability	80%
Mutation probability	20%

5.3. The effect of the population size

We show in Fig. 11 the graphical representation of the evolution of the total fitness function for different population sizes (varying from 5 to 30) and different independent runs (only 5 independent runs are shown). As can be seen in the figure, when the population size exceeds a certain size, the results of total fitness becomes worse. This is due a large population requires more time to transmit good genes to offsprings when only one individual is replaced at once. The threshold of the population size is about 20 individuals.

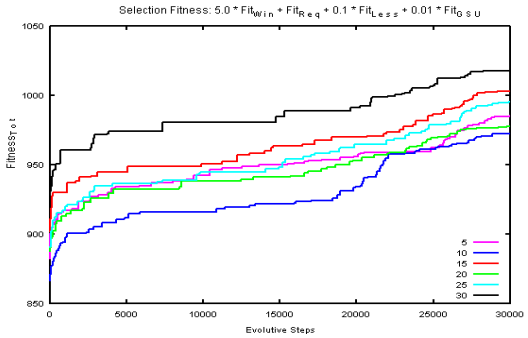

Figure 11: Evolution of total fitness for different population sizes.

Obviously, the larger the population size, the greater is the time needed to processing a population. This is more critical in case of Struggle GA due the computation of similarity function. We study the increase of execution time to see the effect of the increase of population size. As can be seen from Fig. 12, the increase in execution time is almost linear, which confirms the efficiency of the implementation of the hash similarity measures.

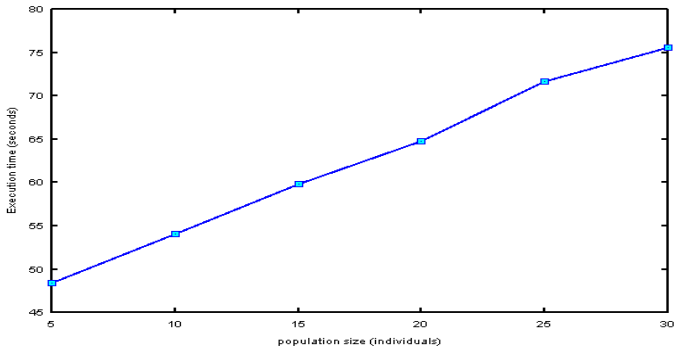

Figure 12: Evolution of execution time for different population sizes (5 independent runs).

5.4. The effect of the mutation probability

Given the importance of the mutation, we studied the variation of the total fitness for different values of the mutation probability. We show in Fig. 13 the graphical representation of the evolution of the total fitness function for different mutation probability values (varying from 1% to 5%) and different independent runs (only 5 independent runs are shown).

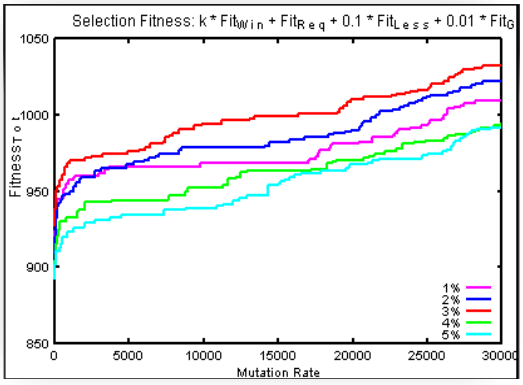

Figure 13: Evolution of total fitness for different mutation probabilities.

5.5. The effect of different similarity measures

The similarity measure is a key factor in Struggle GA as it determines the convergence speed of the algorithm. The experimental study showed that hash similarity function outperforms other similarity measures. On the other

hand, within hash version, the `c key`, which takes into account not only the tasks but also resources, is a better option than `b key`, which considers only the positions of the tasks within the schedule.

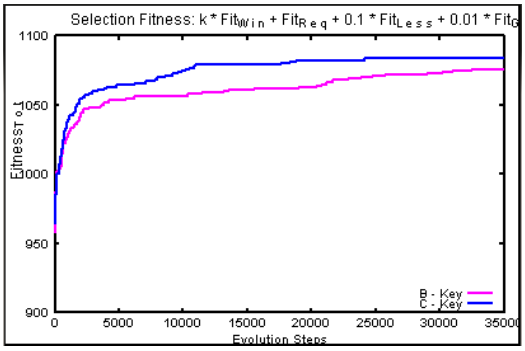

Figure 14: Evolution of total fitness for different hash similarity measures.

5.6. Computational results for different instance sizes

Based on the tuning of parameters presented in the above sections, computational results were obtained for each group of instances (small, medium and large –see Table 5). We give the computational results in Tables 7, 9 and 11, respectively. The standard deviation for each case is given in Tables 8, 10 and 12, respectively. Execution times (averaged) are in seconds. In the table instances are denoted by **I_t_k** , where **t** is the type (**S**–Small, **M**–Medium, **L**–Large size) , and **k** the instance number (**k**=1 . . . 16).

5.7. Evaluation

From the computation results, it can be observed that the Struggle GA computed high quality solutions in terms of different fitness values. This is mainly due the algorithm achieves to maintain a diverse population through the search process. In particular hash based similarity measures contribute to such population diversity even in case of large size instances. Compared to a base GA implementation, the Struggle GA performed much better and consistently on different groups of problem instances.

6. Conclusions and future work

Window time scheduling problems are well-known for their computational complexity due they are over-constrained scheduling problems. One

Table 7: Fitness values for small size instances.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time*
I_S_01	99	70	98.94	66.86	1096.6	39.84
I_S_02	99	70	94.32	65.14	1091.97	40.02
I_S_03	99	57	98.9	69.11	1095.29	39.86
I_S_04	99	78	100	61.5	1098.42	39.66
I_S_05	100	52	97.7	61.1	1103.51	39.83
I_S_06	99	66	96.63	59.01	1093.82	40.91
I_S_07	99	71	94.44	60.79	1092.15	39.71
I_S_08	99	74	100	61.81	1098.02	40.17
I_S_09	99	64	96.59	62.53	1093.62	40.08
I_S_10	100	69	97.85	66.06	1105.41	39.98
I_S_11	99	66	96.7	52.94	1093.83	41.07
I_S_12	98	74	96.74	57.08	1084.71	40.22
I_S_13	99	76	93.68	66.19	1091.95	40.05
I_S_14	100	65	91.95	49.36	1098.95	41.73
I_S_15	98	72	97.78	67.69	1085.65	40.14
I_S_16	81	80	98.31	69.38	917	39.78

Table 8: Mean and std deviation for small size instances.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time*
Mean	99.07	68.27	96.82	61.81	1094.93	40.22
Std deviation	0.59	6.96	2.35	5.53	5.63	0.57

Table 9: Fitness values for medium size instances.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	*Ex. Time*
I_M_01	96.67	74	96.21	44.21	1070.72	104.12
I_M_02	98.67	78	98.5	44.06	1093.4	108.45
I_M_03	98	84	92.59	44.41	1081.44	105.07
I_M_04	98	76	98.53	51.73	1086.65	106.78
I_M_05	99.33	80	96.43	46.71	1098.23	104.15
I_M_06	97.33	88	93.18	45.74	1075.77	104.15
I_M_07	98	68.67	91.3	44.78	1078.62	104.34
I_M_08	99.33	61.33	97.14	51.38	1097.12	104.3
I_M_09	97.33	68	97.76	42.96	1078.32	104.42
I_M_10	99.33	71.33	96.24	40.27	1097.11	107.06
I_M_11	98	80	96.99	49.09	1085.48	104.23
I_M_12	98	75.33	97.79	50.1	1085.83	105.52
I_M_13	98.67	76	97.71	51.36	1092.49	104.77
I_M_14	98.67	57.33	94.74	45.77	1087.59	105.89
I_M_15	99.33	85.33	97.79	49.22	1100.15	105.07
I_M_16	64	92.67	96.55	50.76	746.33	101.53

Table 10: Mean and std deviation for medium size instances.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	Ex. Time
Mean	98.31	74.89	96.19	46.79	1087.26	105.22
Std deviation	0.83	8.54	2.24	3.5	9.05	1.31

Table 11: Fitness values for large size instances.

Instance	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	Ex. Time
I_L_01	97	76.5	95.03	42.8	1073.11	198.53
I_L_02	99	84.5	94.77	39.21	1093.61	202.21
I_L_03	99	87.5	93.64	39.45	1092.79	200.21
I_L_04	98	80	97.18	47.94	1085.65	198.36
I_L_05	98.5	73.5	93.01	39.1	1085.75	199.68
I_L_06	97	76.5	95.05	38.01	1073.09	198.24
I_L_07	97.5	74.5	94.51	35.35	1077.31	198.46
I_L_08	96	76	95.56	36.48	1063.52	200.92
I_L_09	96	80.5	93.18	38.85	1061.62	200.05
I_L_10	98	80	94.54	37.33	1082.91	201.25
I_L_11	96	71	95.05	44.17	1062.6	200.43
I_L_12	99	68.5	98.27	41.31	1095.53	201.72
I_L_13	96.5	76.5	94.41	38.49	1067.45	198.78
I_L_14	98.5	72	97.3	40.21	1089.9	202.92
I_L_15	96	79	96.05	37.18	1064.32	198.28
I_L_16	64	88.5	94.62	53.22	744.01	194.46

Table 12: Mean and std deviation for large size instances.

	Fit_{win}	$Fit_{LessClash}$	$Fit_{TimeReq}$	Fit_{GSU}	Fit_{TOT}	Ex. Time
Mean	97.47	77.1	95.17	39.73	1077.94	200
Std deviation	1.19	5.03	1.5	3.24	12.29	1.55

such problem is that of Ground Station Scheduling, which arises in mission operations for the efficient management of satellite/space missions for communication of spacecrafts with ground stations. The problem is to efficiently allocate a large number of missions (tasks) to a rather small number of ground stations (resources). Given the computational hardness of the problem, meta-heuristics are resolution methods to cope in practice with its efficient resolution. In this paper we have presented the evaluation of Struggle Genetic Algorithms for the problem. Struggle GA is a version of GAs in which an individual replaces another individual of the population only if it has better fitness but additionally should be genetically similar to the individual to be replaced. The aim in Struggle GA is to control the convergence of the GA and avoid premature convergence. This is done by implementing several similarity measures, including some hash-based functions, which achieve high efficiency. We have evaluated the Struggle GA using the STK simulation toolkit. Three groups of different size instances are generated, namely, small, medium and large, aiming to simulate realistic features of the problem. The Struggle GA was able to compute high quality schedules in very short times.

In our future work we would like to implement a parallel version of the Struggle GA to speed up the processing time due computations on populations and genetic operators. This speed up would benefit the proper search time of the GA algorithm.

References

[1] S. Badaloni, M. Falda and M. Giacomin Solving temporal over-constrained problems using fuzzy techniques. *Journal of Intelligent and Fuzzy Systems*, 18(3), 255-265, 2007

[2] L. Barbulescu, A. Howe, J. Watson, L. Whitley. Satellite Range Scheduling: A Comparison of Genetic, Heuristic and Local Search. *Parallel Problem Solving from Nature –PPSN*, VII: 611-620, 2002.

[3] L. Barbulescu, J.-P. Watson, L. D. Whitley and A. E. Scheduling space-ground communications for the air force satellite control network. *Journal of Scheduling*, 7(1), 7-34, 2004.

[4] L. Barbulescu, A.E. Howe, L.D. Whitley, and M. Roberts. Trading places: How to schedule more in a multi-resource oversubscribed scheduling problem. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-2004)*, 227-234, 2004.

[5] L. Barbulescu, A.E. Howe, D. Whitley. AFSCN scheduling: How the problem and solution have evolved. *Mathematical and Computer Modelling*, 43(9-10): 1023-1037, 2006.

[6] S. Damiani, H. Dreihahn, J. Noll, M. Nizette, and G.P. Calzolari. A Planning and Scheduling System to Allocate ESA Ground Station Network Services. *The Int'l Conference on Automated Planning and Scheduling*, USA, 2007.

[7] ESA Science & Technology. http://www.esa.int/

[8] T. Grueninger. Multimodal optimization using genetic algorithms. Technical report. Department of Mechanical Engineering, MIT, Cambridge, MA, 1997.

[9] J. C. Pemberton and F. Galiber. A constraint-based approach to satellite scheduling. In *DIMACS workshop on Constraint programming and large scale discrete optimization*, Eugene C. Freuder and Richard J. Wallace (Eds.). American Mathematical Society, Boston, MA, USA, 101-114, 2000.

[10] A. Sarkheyli, B.G. Vaghei, A. Bagheri. New tabu search heuristic in scheduling earth observation satellites. In *Proceedings of 2nd International Conference on Software Technology and Engineering (ICSTE)*, V2-199 - V2-203, 2010.

[11] W. T. Scherer, F. Rotman. Combinatorial optimization techniques for spacecraft scheduling automation. *Annals of Operations Research*, 50(1):525-556, 1994.

[12] Satellite Tool Kit: http://www.agi.com/products/by-product-type/applications/stk/

[13] N. Zufferey, P. Amstutz, P. Giaccari. Graph Colouring Approaches for a Satellite Range Scheduling Problem. *Journal of Scheduling*, 11(4):263-277, 2008.

[14] J. Sun, F. Xhafa: A Genetic Algorithm for Ground Station Scheduling. International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011, June 30 - July 2, 2011, Korea, 138-145.

[15] F. Xhafa, J. Sun, A. Barolli, M. Takizawa, K. Uchida: Evaluation of Genetic Algorithms for Single Ground Station Scheduling Problem. IEEE 26th International Conference on Advanced Information Networking and Applications, AINA-2012, Japan, March 26-29, 2012, 299-306.